Machine Learning

Priit Järv, TalTech, 2018

Overview

Will cover one example from each:

- Supervised learning
- Reinforcement learning
- Unsupervised learning

Courses ITI0210/ITI0055

Probabilistic models

- Naive Bayes
- Hidden Markov Model

Learning to classify

- Decision Trees
- Neural Networks

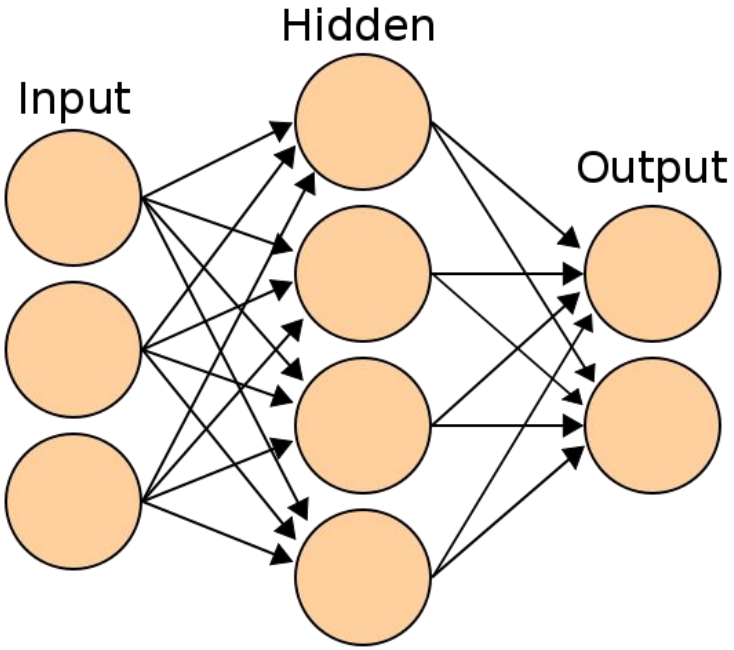
Learning behaviour

- Q-learning

Neural Network

Example of supervised learning

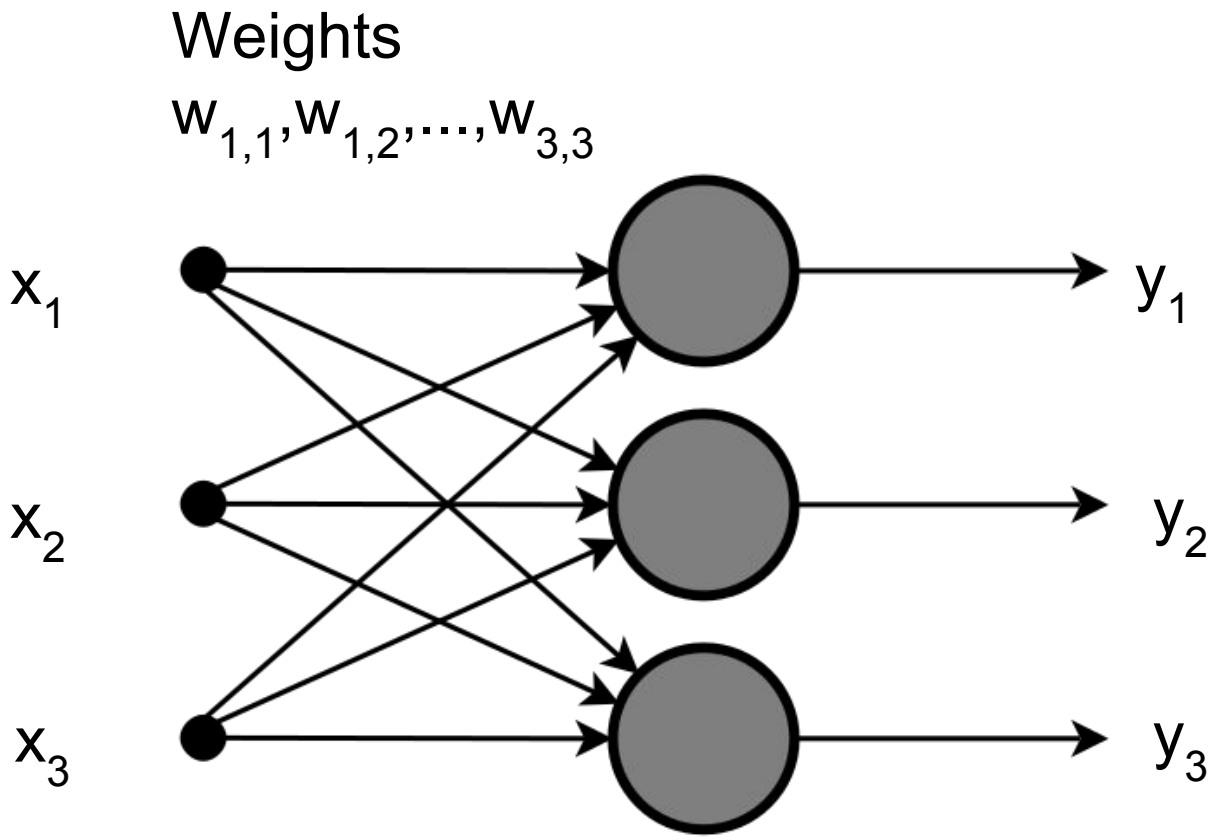
Pixels:



Cat? = 1

Dog? = 0

Neural network: single layer



Neural network: single layer

We want to minimize error on each output y_k

Inputs $x_1 ... x_n$	y_k	Desired output y	Error
Cat pixels	0.78	1	0.22
Dog pixels	0.33	0	-0.33
Another cat	0.96	1	0.04

Usually the error is squared (and called the loss function).

Neural network: single layer

Output of one unit

$$y_k = g(in_k) = g(\sum w_{i,k} x_i)$$

in_k- summed input of unit k

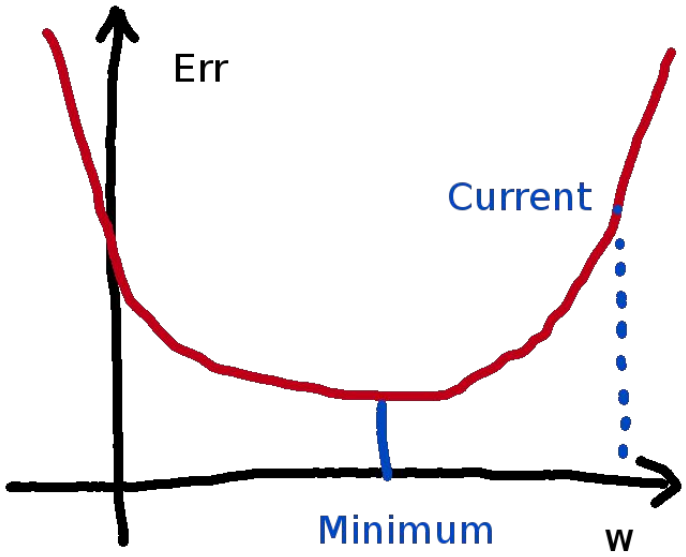
g - activation function

For one training sample, input and desired output are fixed. So error is a function of the weights

$$\mathbf{Err_k} = (\text{Desired } y_k - g(\sum \mathbf{w_{i,k}} x_i))^2$$

Neural network: weight update

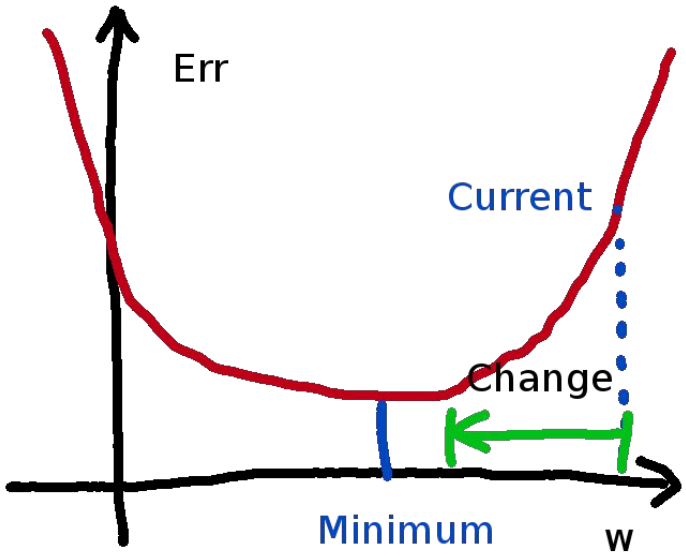
For one training sample, one output and one weight (treat other weights as fixed):



Change the weight to decrease error.

Neural network: weight update

Derivative *Err'(w)* gives the direction of growth, change w in opposite direction.



Update $w_{i,k}$ by

$$\Delta_w = \alpha \, Err_k \, g'(in_k) x_i$$

α - learning rate

Neural network: training process

- For one sample: calculate updates for each unit output, each weight
- Repeat over all training samples
- Do whole process multiple times with different (random) initial weights

Same idea with multi-layer networks, calculate updates layer by layer starting from the back. Called “back-propagation”.

Q-learning

Example of reinforcement learning

Robot in the video learns the behaviour of crawling forward:

https://youtu.be/f2nIKFMyfSg?t=335

Q-learning: states and actions

Crawling robot in the video

- 36 states for the arm (6 angles for each joint)
- 4 actions (two servos, two directions)

Q-learning: states and actions

Table for choosing actions (with some learned values):

Q(s,a)	a_1	a_2	a_3	a_4
s_1	10	9	-5	0
s_2	0	0	-10	0
…	…	…	…	…
s_{36}	0	0	0	3

In state *s*, choose action *a* with highest *Q(s, a)*

Q-learning

Initially, Q can be filled with ‘0’-s.

Choose actions, with some randomness to explore unknown states.

Q-learning

After taking action *a* in state *s* and arriving to *s'*:

$Q(s, a) = Q(s, a) + \alpha(R(s) + \gamma max_{a'} Q(s', a') - Q(s, a))$

R(s) is the direct reward received in previous state. For the crawling robot, change in distance from the sonar.

Q-learning

After taking action *a* in state *s* and arriving to *s'*:

$Q(s, a) = Q(s, a) + \alpha(R(s) + \gamma \max_{a'} Q(s', a') - Q(s, a))$

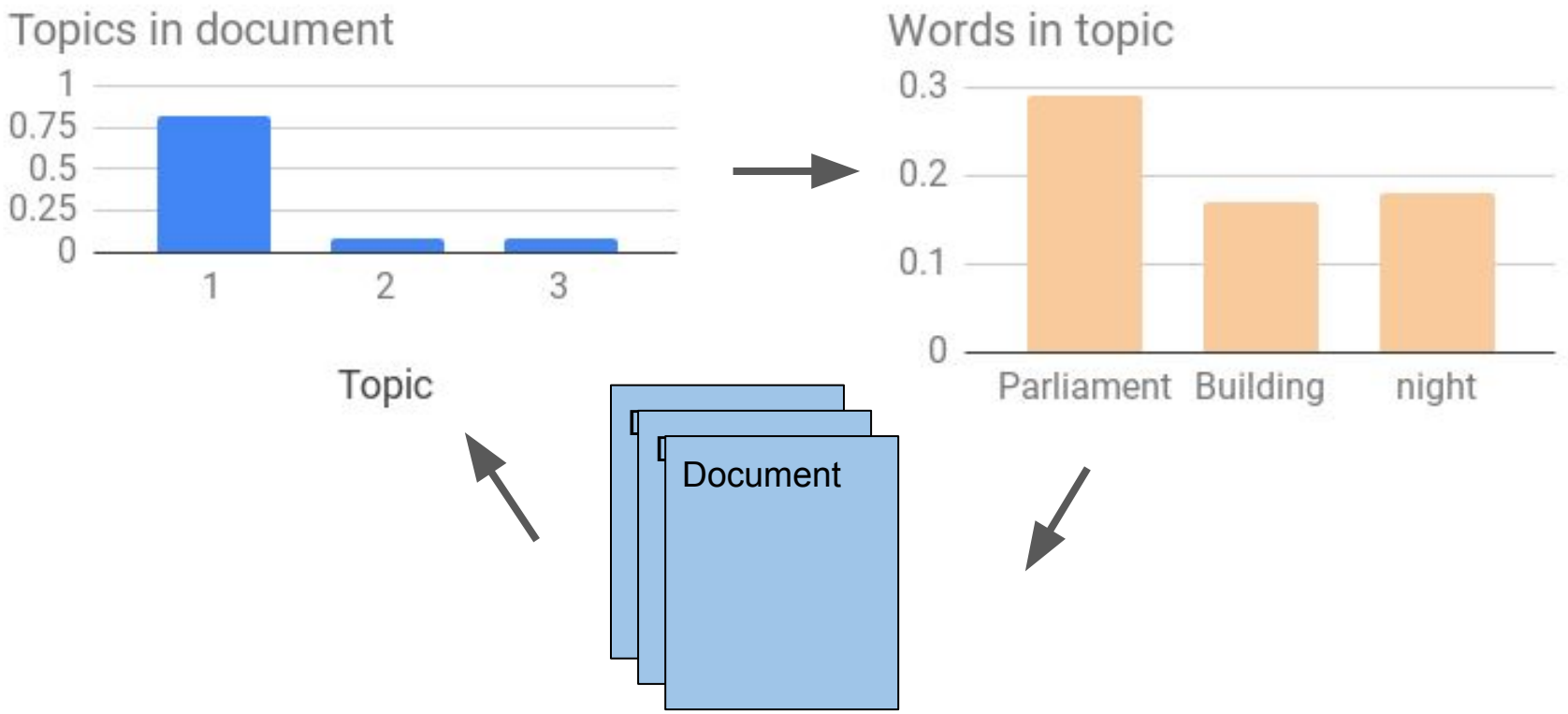
At first, Q-values only change when there is a direct +/- reward (robot has moved).

Then, values propagate to states that lead to rewarding states, etc.

Result: sequence of actions that lead to + rewards.

Topic model (LDA)

Final example: unsupervised learning



Topic model (LDA)

Our “documents”, actually photo captions:

1. “The Hungarian Parliament Building at night”
2. “Hungarian Parliament Interiors”
3. “Budapest Danube at night”
4. “Fisherman’s Bastion by night”

Guess number of topics K=3

Topic model (LDA)

Assign topics to each word in each document (some words ignored)

Parliament	Building	night
1	2	3

One word can belong to different topics in different places (this is OK)

Topic model (LDA)

Counts of topics in documents and topics per word give us the distributions (model)

Parliament	Building	night
1	2	3

	1	2	3	4
1	1	1	1	1
2	1	1	1	1
3	1	0	1	1

	1	2	3
Parliament	2	0	0
Building	0	1	0
night	0	2	1
Interiors	0	1	0
Budapest	0	0	1
...			

Topic model (LDA)

Loop over each document, each word. Draw new topic of the word from the current distributions

Parliament	Building	night
1	2	3

	1	2	3	4
1	1	1	1	1
2	1	1	1	1
3	1	0	1	1

	1	2	3
Parliament	2	0	0
Building	0	1	0
night	0	2	1
Interiors	0	1	0
Budapest	0	0	1
...			

Topic model (LDA): sampling step

Loop over each document, each word. Draw new topic of the word from the current distributions

Parliament	Building	night
1	2	?

	1	2	3	4
1	1	1	1	1
2	1	1	1	1
3	0	0	1	1

	1	2	3
Parliament	2	0	0
Building	0	1	0
night	0	2	0
Interiors	0	1	0
Budapest	0	0	1
...			

Topic model (LDA): sampling step

Probabilities of the new topic using Gibbs sampling (a “Monte Carlo” method)

Parliament	Building	night
1	2	?

Topic 1: 1/2 in doc 1, 0/2 associated to “night”

Topic 2: 1/2 in doc 1, 2/2 associated to “night”

Topic 3: 0/2 in doc 1, 0/2 associated to “night”

Choose randomly, but topic 2 will be most likely

Topic model (LDA): sampling step

New topic assignment changes the distribution

Parliament	Building	night
1	2	2

	1	2	3	4
1	1	1	1	1
2	2	1	1	1
3	0	0	1	1

	1	2	3
Parliament	2	0	0
Building	0	1	0
night	0	3	0
Interiors	0	1	0
Budapest	0	0	1
...			

Topic model (LDA): final result

Actual distributions are based on counts, but a bit trickier (zero counts do not mean 0 probability).

After repeated sampling (only one topic shown)

	1	2	3	4
1	0.82	0.78	0.09	0.09
2	0.09	0.11	0.82	0.11
3	0.09	0.11	0.09	0.80

Topic 1	
Parliament	0.29
night	0.18
Building	0.17